УДК 004.9

Глибовець М. М., доктор фіз.-мат. наук,
Глибовець А.М., кандидат фіз.-мат. наук,
Корень О.М., магістр

Адаптована модель реалізації навчальних матеріалів як компонентів повторного використання

В роботі проаналізовані проблеми створення електронних навчальних пакетів(НП) та наведено можливі шляхи їх розв'язання. Описано розроблену адаптаційну об'єктну модель НП разом з концептуальнюї моделлю життєвого циклу НП та інструментальний засіб створення SCORM-сумісного НП.

Ключові слова: навчальний пакет, метадані, стандрат,специфікація.

Glybovets M. M., D.Sci.(Phys.-Math.)
Glybovets A.M.,
Koren O.M.

Adapted implementation model of teaching materials as reuse components

Present work analyzes the problems of electronic educational packages(EP) creation and provides possible ways of solving them.

Describes the developed adaptation object model of EP, along with conceptual model of EP life cycle and instrumental case of SCORM – compatible EP creation.

Key Words: educational package, metadata, standard, specification.

*E-mail: glib@ukma.kiev.ua, ankoren@gmail.com

Статтю представив д.ф.-м.н., проф. Анісімов А.В.

Вступ

На сьогоднішній день розробниками електронних навчальних матеріалів ще не вирішено в повному обсязі цілий ряд задач. Природно ці задачі можна розділити на три основні категорії: проблема повторного використання, складність опису мета даних, педагогічна ефективність.

Значну кількість розробок та специфікацій було присвячено саме можливості навчальних пакетів (НП) повторно використовуватись у різних навчальних середовищах [1]. Серед них виділяються стандарт IMS Content Packaging [2] та стандарт опису навчального об'єкту IEEE LTSC (Learning Technology Standards Committee) LOM (Learning Objects Metadata) [3]. Власне лише ці два стандарти дозволяють легко імпортувати навчальний об'єкт до певного навчального середовища, а також дозволяти користувачам знаходити його.

Висвітленню ще одного підходу до розв'язку данної задачі і присвячені матеріали цієї статті.

Як радить [4], для того щоб навчальний матеріал можна було скомпонувати як об'єкт повторного використання він має задовольняти двом основним властивостям: *когерентність* (cohesion) – навчальний об'єкт має бути одно направленим, і служити єдиній меті; *незалежність* (coupling) – кожен навчальний об'єкт має бути самодостатнім і ні в якому разі не мати зовнішніх залежностей.

Для вирішення проблеми суто повторного використання навчальних об'єктів застосовують давно перевірені та надійні прийоми з розробки програмних комплексів. Нижче наведемо лише кілька вдалих прийомів, яких ми дотримуватимемося під час проектування інструментальних засобів для компонування навчальних пакетів та для архітектури власне структури пакетів [5]:

- *Дотримання мінімальної зв'язності навчального пакету*. В ідеалі, скомпонований навчальний пакет не повинен жодним чином залежати від ресурсів чи метаданих, що знаходяться поза його середовищем.
- *Максимальна когерентність пакету*. Навчальні матеріали, що містяться у пакеті повинні стосуватися лише певної чіткої тематики, або відповідати одній меті. Власне дотримування даного принципу передбачає собою дотримання вимоги модульності у програмуванні.
- *Чітка декомпозиція частин пакету*. Структура маніфесту навчального пакету дозволяє користувачеві досягти чіткого представлення частини або повного курсу у вигляді відповідної ієрархії організацій та ресурсів.

Наразі, задача пошуку необхідних навчальних матеріалів досить ускладнена. Більшість серверів підтримують лише повнотекстовий пошук. Метадані ж дозволяють більш точно проводити знаходження потрібних результатів шляхом надання додаткових можливостей для детального опису навчального ресурсу. Проте і метадані створюють проблеми, а саме [6]:

- *Погана підтримка пошукових серверів*. Більшість пошукових сервісів загального призначення не підтримують нотацію метаданих, їх підтримують лише спеціалізовані пошукові машини навчальних середовищ. Але ця проблема

вирішиться з часом, коли метадані набудуть значного поширення.

• *Відсутність власне метаданих*. Таке трапляється коли автори навмисно не надають ніякої додаткової інформації. Мотивується таке рішення відсутністю явної ефективності, тому заповнення метаданих як правило відкладається на потім. Насправді ж наявність метаданих не лише покращить шанси навчального пакету на те щоб бути знайденим, а й полегшить інтеграцію або імпорт пакету до розподілених навчальних середовищ.

• *Багато різних стандартів*. Розробники мають у своєму розпорядженні значну кількість рекомендацій та специфікацій метаданих для навчальних матеріалів. Потрібно розв'язувати проблему цніфікації вживаної термінології. Навіть, якщо специфікації IMS та ARIADNE базуються на IEEE LTSC LOM, і здаються подібними, вони все одно відрізняються на більш детальному рівні, щопотребує значного часу на їх вивчення. Багато користувачів плутаються у термінах, або потребують створення власного словника позначень елементів метаданих.

В [6] пропонується вирішення поставлених проблем використання метаданих автоматитизацією «витягання» даних з певного типу документів. На практиці, за наявності спеціального API (Application Programming Interface) з кожного типу документа, малюнка чи мультимедійного файлу (звук, відео, анімація) можна отримати інформацію про автора, службову інформацію про середовище створення файлу (наприклад необхідна роздільна здатність для перегляду інтерактивної презентації) тощо. У нашому випадку вдалим рішенням буде просто надання авторові навчального пакету зручного інструменту для ведення метаданих.

Окрім проблеми створення НП у вигляді компонента повторного використання є ще одна проблема використання власне пакету у процесі навчання – проблема педагогічної ефективності. Раніше, ми заявили, що для того щоб пакет був максимально повторно використовуваним, ми дотримуємось принципу незв'язності та деконтекстуалізації. Проте як відомо, максимальну ефективність від навчання ми отримуємо лише за умови, коли учень вирішує реальні задачі. У технічному аспекті це означає, що ресурси навчального пакету мають посилатися на зовнішні ресурси для створення реального контексту задачі. Проте це суперечить нашим принципам побудови модульності пакету. Отже маємо певну суперечність, яка є нерозв'язною в певному сенсі. Але така ситуація не повинна в жодному разі заводити розробника в глухий кут. Вирішенням є підтримка балансу у навчальному пакеті контекстно-залежних та контекстно-незалежних ресурсів, а також надання можливості швидко встановити контекст для навчальних матеріалів безпосередньо після імпорту до розподіленого навчального середовища.

Основною задачею даної роботи є реалізація розв'язання поставлених вище проблем у вигляді адаптаційної моделі налаштовуваного компонента повторного використання.

1. Модель реалізації навчальних матеріалів

У різних навчальних середовищах, моделі представлення навчальних матеріалів можуть бути різними. В ILIAS [3], наприклад, навчальні матеріали та ресурси можуть бути представлені як у вигляді курсу, так і у вигляді електронної бібліотеки або просто директорії ресурсів (media pool). Але, усі ці ресурси при експорті до інших систем, компонуються в певний НП, який має універсальну структуру.

Розробники чи команда розробників навчального середовища намагаються створити власний формат пакування навчальних ресурсів та власний формат опису даних ресурсів. З часом, більшість з них зійшлись на тому, що для універсального опису матеріалів найкраще розміщувати інформацію в одному файлі у вигляді метаданих, що і підтримує давно відомий принцип мета програмування [5].

Проте і тут виникла проблема, яку ми вже зазначили вище. Якщо кожен розробник підтримує лише свій тип метаданих – це унеможливлює імпорт/експорт навчальних матеріалів до систем сторонніх розробників. Ця проблема на сьогодні також вирішена стандартом IMS CP [7], а також рекомендаціями та вимогами SCORM (Shared Content Reference Model)CAM [8].

Для багатьох розробників рекомендована структура навчального пакету (content package) є очевидною, прозорою та зручною, проте для непрофесійного користувача структура навчального пакету, поняття комбінації (sequencing) чи агрегації є дещо віддаленими поняттями від безпосередньо тих переваг які очікуються. Кожне навчальне середовище вирішує цю проблему по-своєму: Moodle окремо виділяє модуль SCORM-курсів, які є імпортованими навчальними пакетами; ILIAS вдало представляє навчальний пакет як навчальний курс з розділами посиланнями та зовнішніми ресурсами.

Для прозорості представлення навчального курсу було розроблено власну модель, яку можна застосовувати безпосередньо для інструментальних засобів по створенню НП.

1.1. Адаптована модель навчального пакета

Першим кроком адаптації існуючих моделей та специфікацій є уточнення понять та термінів, що використовуються. Для цього спеціалізовані терміни максимально наближені до прозорості використання. Ідея полягає у створенні враження, що навчальний пакет представляє собою інтерактивну книжку, яка сама перегортає сторінки. Проте завданням розробленого інструментального засобу не є деагрегація та представлення вмісту навчальних пакетів згідно інструкцій метаданих. Розглянемо та порівняємо поняття, що декларуються SCORM та IMS, та введені нами для відображення ієрархічної структури НП (таблиця 1).

Таблиця 1. Відображення понять специфікації у контексті застосування.

Офіційні поняття специфікацій	**Відображені поняття**
Пакет вмісту (Content Package)	Навчальний пакет
Маніфест (Manifest)	Структура навчальних матеріалів
Метадані (Metadata)	Анотація навчального пакету
Організації (Organization)	Розділ
Предмет (Item), контейнерний тип[1]	Підрозділ
Предмет (Item)	Сторінка, навчальний об'єкт[2]
Ресурс (Resource)	Ресурс

Дані поняття використовуються як позначення та пояснення до застосування інструментального засобу компонування навчальних матеріалів. Розглянемо кожен елемент навчального пакету більш детальніше та сформуємо підмножину тих елементів, які будуть надані у розпорядження інструментального засобу. При цьому ми намагатимемося водночас мати таку підмножину специфікації SCORM-сумісного навчального пакету, яка буде потім сприйматись багатьма існуючими навчальними середовищами та іншими програмними засобами.

1.2. Застосування стандарту IMS Content Package та SCORM CAM

Безумовно, концептуальна модель, що була представлена стандартом IMS Content Package, є дуже зручним засобом для універсального представлення навчальних пакетів. Специфікація вимог SCORM CAM адаптувала її для використання у кінцевих системах, додавши деякі власні елементи та елементи метаданих LOM. Наша задача полягає у кінцевій адаптації моделі, яку можна використовувати безпосередньо як об'єктну модель до реалізації модулів навчальних середовищ та інструментальних засобів.

Файл Обміну Навчальним Пакетом (Package Interchange File, далі - PIF) – є контейнером навчальних матеріалів повторного використання у вигляді архіву. PIF містить файл маніфесту зі спеціальним іменем *imsmanifest.xml*, а також усі необхідні файли представлення та ресурси, задекларовані у маніфесті. Власне SCORM рекомендує представлення PIF як єдиного файлу, з використанням архіватора PKZip 2.04g, який відповідає вимогам RFC1951 [9].

Нижче розглянемо елементи маніфесту навчального пакету та специфікуємо власну підмножину, необхідну для SCORM-сумісності.

Елемент <*manifest*>

Він є елементом повторного використання, який має контейнерний тип і інкапсулює метадані, організації та посилання на ресурси. <***manifest***> є кореневим елементом у файлі маніфесту. Якщо даний елемент зустрічається усередині кореневого елементу, то вся підмножина називається суб-маніфестом і може бути використана з метою агрегації, деагрегації чи повторного використання частин навчального пакету.

*Тип даних***:** даний елемент має контейнерний тип і не має безпосереднього значення.

---

[1] Згідно специфікації SCORM CAM елемент Item може бути як листком в ієрархічному дереві пакету, так як і контейнером для інших елементів Item. В даному випадку йдеться про Item, як контейнер.

[2] Якщо типом Item є певний текстовий документ, то в контексті навчального пакету його краще розглядати як сторінку. Якщо ж Item посилається на ресурс мультимедіа (скажімо презентацію), то краще представити елемент у вигляді навчального об'єкту.

*Атрибути***:**

Назва	Обо'язковий	XML Тип	Опис
identifier	Так	xs:ID	Унікальний ідентифікатор елемента, надається автором чи інструментальним засобом.
version	Ні	xs:string	Ідентифікатор версії, вказується, якщо існує кілька версій маніфестів з однаковим ідентифікатором.
xml:base	Ні	xs:anyURI(Unified Resource Identifier)	Відносний шлях до локальних ресурсів, на які посилається маніфест.

*Внутрішні елементи***:** <metadata>, <organizations>, <resources>, <manifest>

Аналогічно описуються і елементи: **<***metadata***>**, <*location*>,<*organizations***>.**

Елемент <*metadata*> є контейнером для метаданих, по питанням що стосуються навчального пакету (наприклад, можливості агрегації пакету). Елемент є кореневим. Це означає, що усі елементи метаданих мають бути розміщені у ньому. Розміщення метаданих у навчальному пакеті відбувається на основі розширень XML. SCORM рекомендує як мінімум розміщувати інформацію про навчальний пакет згідно стандарту LOM. Інші метадані також можна розміщувати. Механізмів ведення метаданих може бути кілька: безпосередньо у маніфесті, або у зовнішньому файлі, на який попередньо описане посилання елементом <*location*>. Елементи опису LOM можна включати до секції метаданих шляхом розширень іменного простору (namespace) XML. Елемент <*organizations*> є контейнером для однієї або більше організацій. Він представляє елемент ієрархічної структури навчального пакету. Власне термін «організація» є спеціальним і може означати урок, сторінки, модуль тощо. У нашому випадку організація є розділом. В термінах стандарту IMS Simple Sequencing [10] організація називається завданням (activity).

Описовий елемент <*title*>, служить для учня чи викладача описом певної організації НП. В залежності від опису, вона може бути курсом, уроком, модулем тощо. Елемент **<***item***>** описує ієрархічну структуру матеріалів. У нашому випадку елемент означає фрагмент певного уроку чи курсу і служить для подання матеріалів у вигляді дерева. Фрагмент може бути як кінцевим атомарним елементом, так і контейнером для інших елементів даного типу.

Елемент <*adlcp:timeLimitAction*> описує дію, що має бути виконана, якщо максимальний час на виконання дії, пов'язаної з фрагментом НП було вичерпано. Даний елемент може мати лише ті фрагменти (елемент <*item*>), які безпосередньо посилаються на ресурс. Елемент є задекларованим розширенням специфікації IMS CP. Обов'язок перевірки часових обмежень має брати на себе навчальне середовище. Якщо ресурс є інтерактивним (застосування з тестовими завданнями), то він і бере на себе це зобов'язення.

Елемент <*adlcp:dataFromLMS*> містить у собі інформацію, що може бути корисна на етапі ініціалізації ресурсу, на який посилається елемент <*item*>. Ця інформація є прозорою для навчального середовища. Якщо розробнику необхідно передати певні параметри для безпосередньої ініціалізації ресурсу, то їх необхідно передавати через вже зазначений атрибут *parameters* елементу <*item*>.

Елемент <*adlcp:completionThreshold*> задає певне порогове значення проходження навчального підрозділу, що може бути використане навчальним об'єктом. Даний елемент стосується лише тих елементів <*item*>, які посилаються на ресурси.

Елемент маніфесту <*resources*> є контейнером для елементів посилань на ресурси. Відомі специфікації не вказують в якому порядку чи ієрархії повинні слідувати елементи ресурсів.

Якщо елемент <item> посилається на ресурс, то реальний файл ресурсу знаходиться за посиланням та доставляється безпосередньо учневі всередині навчального середовища. Згідно SCORM, цей елемент має відповідати наступним вимогам. Атрибут *type* має бути зі значенням *webcontent*. Атрибут *adlcp:scormType* набуває значень лише *sco* чи *asset*. Атрибут *href* є обов'язковим.

Елемент <*file*> надає безпосереднє посилання на файл ресурсу. Для кожного файлу необхідно вносити один елемент даного типу, формуючи таким чином множину файлів для конкретного ресурсу. Даний елемент має посилатись на файли, які знаходяться локально, тобто фізично розташовані в одному навчальному пакеті з маніфестом.

Елемент <*dependency*> символізує залежність певного ресурсу від іншого ресурсу, що

знаходиться в даному маніфесті. Останній може містити кілька файлів.

Описані вище елементи формуватимуть підмножину SCORM- сумісного навчального пакету, що буде генеруватись розробленим інструментальним засобом. До даної підмножини не було включено елементів, що відповідають безпосередньо за процес представлення навчального матеріалу, а саме елементів IMS SS (Simple Sequencing), а також IMS Learning Design [11]. Нас цікавить виключно та мінімальна підмножина елементів маніфесту, за допомогою якої можна побудувати навчальний пакет, а також ефективно проводити його пошук та доставку.

Далі розглянемо більш загальну картину життєдіяльності навчального пакету в контексті навчального процесу, середовища та акторів.

1.3. Компоненти життєвого циклу навчальних матеріалів: створення, збереження, використання

Попередня модель стосувалась лише побудови навчального пакету як компонента повторного використання. Розглянемо тепер загальну модель взаємодії навчального пакету в контексті використання всередині навчального середовища та його повний життєвий цикл (life cycle). Для кожного компонента життєвого циклу наведемо також його коротку специфікацію та рекомендації до реалізації чи вибору компонента серед готових програмних засобів. Слід також зазначити, що дана модель є спрощеною для покращення сприйняття, тому в ній відсутні такі актори як, вчитель (tutor), які є дуже важливими в контексті процесу навчання, а не в циклі використання навчальних матеріалів.

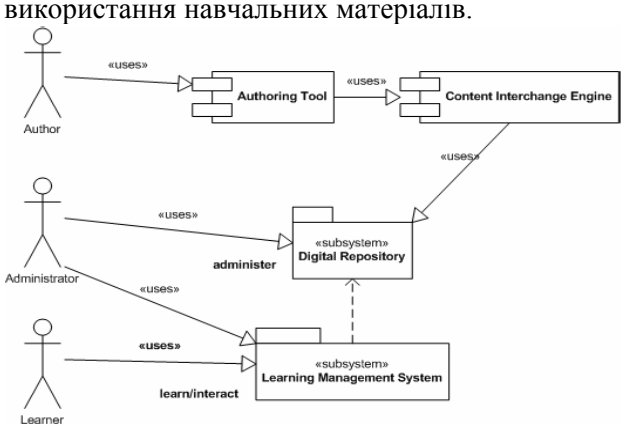

Рис. 1 Компоненти життєвого циклу навчальних матеріалів.

Розглянемо кожен з компонентів та акторів, що зображені на рис. 1 детальніше. Нижче наведено опис акторів, що беруть участь у життєвому циклі навчальних пакетів: *Автор (Author)* – представляє будь-яку особу (студент, викладач, асистент), що має деяку підбірку навчальних матеріалів та бажає оформити її у вигляді навчального пакету; *Адміністратор (Administrator)* – людина з технічного персоналу, що відповідає за надійність роботи репозитарію навчальних матеріалів та навчальне середовище; *Учень(Learner)* – особа, яка користується НП у процесі навчання.

Окрім акторів маємо також такі компоненти:

- *Інструментальний засіб створення навчальних пакетів (authoring tool)* – розуміється спеціалізоване програмне забезпечення, що надає можливість *авторові* створювати SCORM та IMS сумісні навчальні пакети. Метою даної роботи є опис розробки такого засобу.
- *Сервіс обміну навчальними матеріалами (content interchange engine)* – під сервісом розуміється веб-базоване програмне забезпечення, що виконує функції імпорту та експорту навчальних пакетів, пошук та доставку навчальних пакетів за метаданими, збереження навчальних пакетів у репозитарії. Даний компонент тісно пов'язаний з репозитарієм навчальних ресурсів і для ефективності послуг пошуку має відповідати стандарту IMS Metadata.
- *Репозитарій навчальних ресурсів (digital repository)* – репозитарій є кінцевою точкою для збереження навчальних пакетів. Програмне забезпечення, що виконує дані функції повинно бути сумісним з вимогами IMS DRI (Digital Repository Interoperability). Репозитарій є сервіс-провайдером для усіх компонентів, що працюють з навчальними пакетами.
- *Навчальне середовище (learning management system)* – це середовище представлення (rendering) навчальних пакетів та система керування навчальним процесом. Середовище знаходиться в залежності від репозитарію навчальних ресурсів і взаємодіє з ним на рівні імпорту/експорту навчальних пакетів. Правильно організоване навчальне середовище має відповідати специфікаціям IMS LIP (Learner Information Package), IMS Learning Design, IMS SS. Для тестового провадження розробленого в даній роботі інструментального засобу Content Packager пропонується використовувати системи ILIAS або Moodle.

2. Реалізація інструментального засобу "Content Packager"

Опишемо особливості процесу реалізації інструментального засобу Content Packager, мета якого надати авторам засіб легкої та прозорої компоновки SCORM-сумісних навчальних пакетів. Даний засіб було розроблено на факультеті інформатики НаУКМА для налагодження ефективного обміну знаннями.

2.1. Огляд оновлених вимог до реалізації

Проблема створення подібного інструментального засобу була поставлена з часів створення першої версії специфікації IMS CP. Проте згодом виявилось, що для коректної реалізації також необхідно підтримувати багато інших специфікацій IMS та IEEE LTSC. Для спрощення обсягу робіт задачу створення універсального НП, було зведено до надання авторові можливості відобразити певний хаотичний набір навчальних матеріалів у чітку ієрархічну структуру з можливістю надання мінімально необхідного набору метаданих. Отже головні функції розроблюваного інструментального засобу зводяться до наступного: створення деревоподібної ієрархічної структури розділів навчального пакету, його підрозділів та сторінок; редагування властивостей кожного вузла дерева, та специфікація його типу; редагування метаданих підрозділу; імпорт ресурсів та створення посилань на ресурси окремих сторінок навчального пакету; компіляція навчального пакету; збереження тимчасового робочого маніфесту навчального пакету.

Розглянемо технічні вимоги до реалізації інструментального засобу. Дане програмне забезпечення має бути виконаним з візуальним віконним інтерфейсом, хоча і допускається створення command-line версії для підтримки автоматизованої збірки пакетів. Обов'язковою вимогою є інтуїтивність інтерфейсу користувача, що має мінімальний досвід роботи з комп'ютером та операційною системою.

Інструментальний засіб планується використовувати для компонування навчальних матеріалів, які будуть використовувати в різних навчальних середовищах, зокрема ILIAS та Moodle. При розширенні кількості навчальних середовищ або введення спеціалізованих репозитаріїв навчальних ресурсів, контекст застосування даного інструментального засобу не зміниться.

2.2. Реалізація

Процес розробки інструментального засобу було вирішено вести з частковим застосуванням технології XP, зокрема щодо опису функціональних вимог. Останні були описані у вигляді карток і згруповані за типом функціональності: формування структури, особливості візуального інтерфейсу, збереження навчального пакету. Далі було проведено огляд технологій, які б підходили для заданої структури класів

Оскільки на сьогоднішній день значна частина користувачів, надають перевагу роботі в UNIX-подібних системах, однією з головних не функціональних вимог, була підтримка мультиплатформенності. Для виконання даної вимоги було обрано платформу Java, як найбільш зручну для виконання кросплатформенних застосувань.

Наступним кроком було прийняття рішення щодо вибору каркасної моделі для реалізації ініціалізації та зв'язування головних класів інструментального засобу. Для цього було обрано легковісний контейнерний фреймворк Springframework [12], який реалізує патерн «ін'єкції залежностей» (Dependency Injection [13]). Відомий факт, що використання даної технології пропагує використання усталених патернів проектування застосування. Детально розглядати їх ми не будемо, проте тенденції легко відслідковуються на діаграмі класів та інтерфейсів.

Для реалізації прототипу інтерфейсу та власне інтерфейсу застосування було обрано технологію Thinlet [14]. Дана розробка є набором інструментів та бібліотек для створення «легких» інтерфейсів, що можна застосовувати на мобільних пристроях. Слід також зазначити, що особливою перевагою даного інструментального засобу є можливість опису набору компонентів та поведінки інтерфейсу за допомогою декларацій в спеціальному XML файлі.

Для виконання завдань збереження маніфесту навчального пакету та проміжного робочого маніфесту, а також читання було застосовано технологію та інструментальний пакет Xerces [15].

Діаграма класів представлена на рисунку 2.

Візуальний інтерфейс користувача обслуговується класом *ApplicationMain*, який водночас виконує роль *Controller* у моделі MVC [16], яку реалізовано при створенні інструментального засобу. Застосування запускається класом *ApplicationLauncher*, що

створює контекст для роботи контейнера *Spring*, та активує *ApplicationMain*.

В свою чергу *ApplicationMain* створює візуальні компоненти для подальшого використання, що відповідають за різні функціональні можливості інструментального засобу: *ImportWizard* – діалогове вікно опцій імпорту навчальних пакетів; *StructureEditorArea* – вікно редагування структури навчального пакету, містить компонент типу дерева для відтворення ієрархічного характеру структури; *PropertyEditor* – область редагування властивостей елементів дерева (організацій, маніфесту, ресурсів тощо); *MetadataEditor* – область редагування властивостей метаданих елементів дерева.

Рис. 2 Класи візуального інтерфейсу.

Для відображення візуального представлення ієрархічної структури навчального пакету, клас *ApplicationMain* використовує такі класи рівня структур даних (рис. 3.).

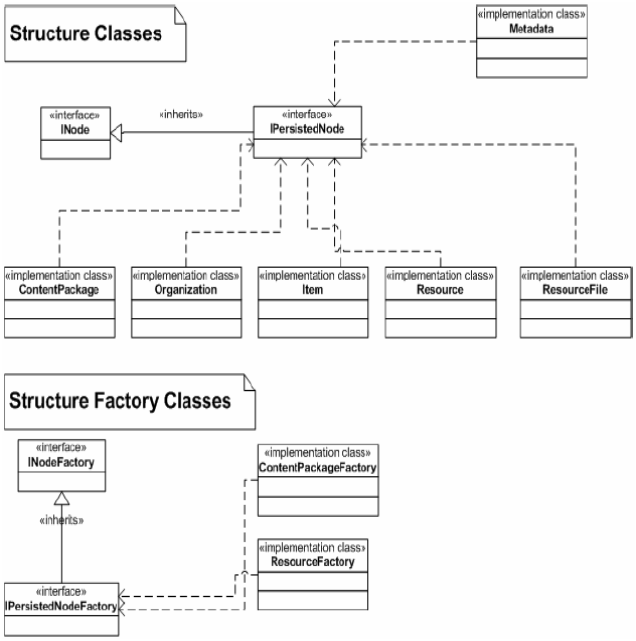

Рис. 3 Класи структур та factory-класи.

Розглянемо інтерфейси, що реалізуються класами: *INode* – надає засоби ідентифікації елемента структури, його ім'я та список атрибутів; *IPersistedNode* – наслідує *INode* та надає можливості виконання операцій завантаження/збереження/видалення для елементів; *INodeFactory* – створює екземпляри класів *INode* за певними параметрами; *IPersistedNodeFactory* – відтворює екземпляри класів *INode*, попередньо збережених в певному вигляді. В нашому випадку, таким типом збереження є XML.

Дані інтерфейси реалізуються класами, які виконують такі функції: *Metadata, ContentPackage, Organization, Item, Resource, ResourceFile* – це реалізації структур даних для збереження даних про відповідні елементи навчального пакету; *ContentPackageFactory, ResourceFactory* – factory-методи для створення вищезазначених структур. Дану функціональність було розділено у два класи, оскільки *ResourceFactory* має специфіку роботи власне з файлами ресурсів та внутрішньою ієрархією каталогів навчального пакету.

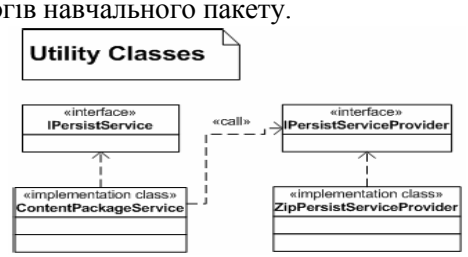

Рис. 4 Утилітні класи та інтерфейси.

Утилітні класи (рис.4.) забезпечують відображення структури маніфесту та ресурсів навчального пакету в безпосередньо єдиний *.zip* файл, а також надають можливість імпорту інших навчальних пакетів: *IPersistService* – інтерфейс провадить функції для реалізації правил збереження абстрактної структури; *IPersistServiceProvider* – інтерфейс провадить функції для збереження абстрактної структури, що представлена у довільному вигляді; *ContentPackageService* – клас реалізації, в якому задані правила та порядок формування навчального пакету; *ZipPersistServiceProvider* – утилітний клас, який надає засоби створення *.zip* архіву і розміщення маніфесту та ресурсів навчального пакету у ньому.

Зазначимо, що тут приведено лише основні класи, та відношення наслідування. Інші класи, що були розроблені в інструментальному засобі є більше утилітними і не виконують безпосередньої ролі в реалізації функціональності побудови навчального пакету.

Інтерфейс користувача підтримує наступні головні сценарії користувача, що працює з

навчальним пакетом: створення нового проекту навчального пакету, розміщення тимчасового сховища для ресурсів та маніфесту[3]; побудову структури навчального пакету у вигляді розділів, підрозділів та сторінок; заповнення метаданих для маніфесту, всього навчального пакету, як навчального об'єкту, та безпосередньо кожного елемента структури; імпорт ресурсів та створення посилань на ресурси кожної сторінки навчального пакету; збереження тимчасового проекту навчального пакету; компіляція навчального пакету в окремий *.zip* файл; імпорт відкомпільованого навчальногопакету або пакету стороннього розробника та продовження роботи над ним.

Висновок

В роботі було окреслено коло проблем, пов'язаних з використанням НП та наведено можливі шляхи їх вирішення. Коротко описано розроблену адаптаційну об'єктну модель НП, концептуальну модель життєвого циклу НП та розроблений інструментальний засіб створення SCORM-сумісного НП. Одна з концепцій повного відображення навчальних пакетів у вигляді набору об'єктів представлена у [17].

Надалі планується провести такі вдосконалення розробленого інструментального засобу як введення можливості створення локалізаційних пакетів, автоматичний імпорт схем даних для структури навчального пакету та мета даних, підтримка інших засобів архівації навчального пакету. Даний засіб було розроблено за досить гнучким підходом, який дозволяє вносити до нього автоматичні зміни у разі зміни функціональних вимог, вимог стандартів навчальних пакетів чи метаданих. Оскільки усі компоненти конфігуруються всередині контейнера Spring, написання нових модулів не складе проблеми за умови підтримки існуючих інтерфейсів.

Основною проблемою для розробників лишається складність реалізації структури маніфестів та інформації про навчальний об'єкт в розподілених навчальних середовищах та інструментальних засобах. Особливо це стосується реалізації, таких складних аспектів IMS як Simple Sequencing, Learning Design, Question and Test Interoperability тощо. Тому перед безпосереднім етапом проектування майбутнього застосування необхідно розробити адаптацію рекомендованих моделей для конкретних потреб та функціональних вимог.

[3] За задумом, до того як проект навчального пакету є опублікованим, маніфест та файли ресурсів знаходяться фізично на диску в певній директорії.

Список використаних джерел

1. Глибовець М.М., Постніков О.С. Використання мобільного агента при пошуку в розподілених репозиторіях навчальних об'єктів, Вісн. Київ. нац. ун-ту ім. Т. Шевченка. Сер. : Фіз.-мат. Науки. - 2004. - Спец. вип. - С. 128-130.
2. www.imsglobal.org/content/packaging/
3. www.ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf
4. Boyle, T. "Design principles for authoring dynamic, reusable learning objects". Australian Journal of Educational Technology, 19 (1), 46-58
5. Перевозчикова О.Л. Основи системного аналізу об'єктів і процесів комп'ютеризації. - Київ, ВД «КМ Академія», 2003.-432с.
6. www.sonntag.cc/fileadmin/Downloads/.../Metadata_in_E-Learning.pdf
7. www.imsglobal.org/content/packaging/.../imscp_infov1p1p4.html
8. www.edu.ru/db/portal/e-library/00000053/SCORM-2004.pdf
9. IETF RFC 1951 DEFLATE Compressed Data Format Specification version 1.3, May 1996 Available at: http://www.ietf.org
10. IMS Simple Sequencing Behavior and Information Model v1.0 Final Specification, IMS Global Learning Consortium, Inc., http://www.imsproject.org
11. Rob Koper, Bill Olivier, Thor Anderson. IMS Learning Design Best Practice and Implementation Guide. www.imsglobal.org/learningdesign/imsld_bestv1p0.html
12. Spring framework Lightweight Container. http://www.springframework.org/about
13. M. Fowler. Inversion of Control Containers and the Dependency Injection pattern. 2004. http://www.martinfowler.com/articles/injection.html
14. Thinlet GUI Toolkit. http://thinlet.sourceforge.net
15. Xerces2 Java Parser. http://xml.apache.org/xerces2-j
16. Design Patterns: Model-View-Controller. Sun Microsystems. 2002. http://java.sun.com/blueprints/patterns/MVC.html
17. Permanand Mohan, Christopher Brooks . Engineering a Future for Web-based Learning Objects. www.informatik.uni-trier.de/.../Brooks:Christopher_A=.html

Надійшла до редколегії 14.01.2010